

VK Newsletter


Software

Development is

subjective

 [Blog](#) |  [Newsletter](#) |  [Founders Guide](#) |  [CTO Guide](#)

Vadim Kravcenko - *12th April 2022*

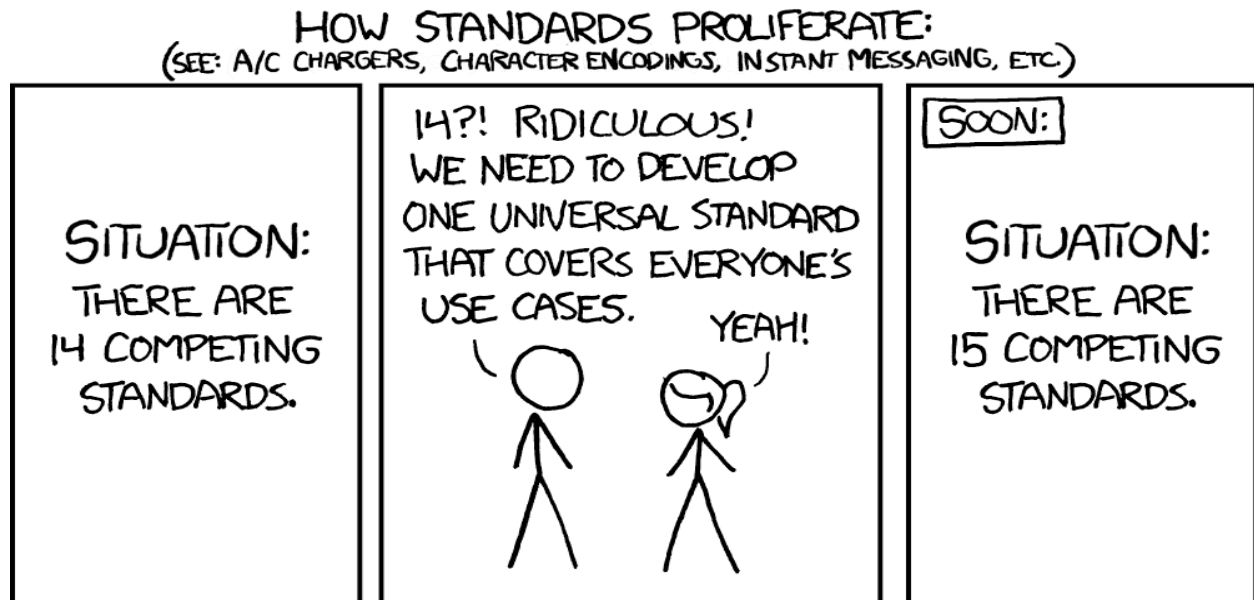


Opinionated Practices	3
Opinionated Frameworks	4
Opinionated Products	6
Opinionated Processes	8
Opinionated Conclusion	9

Most of you are familiar with the feeling of joining a new company and have that urge to rewrite everything. Seeing the blasphemy that your new team members committed a few years ago makes your eyes hurt. Of course, you know better, and you would follow the best practices when developing that feature. Right?

Probably. But over the years, I learned that the problem with “best practices” is that they’re very subjective. Each company you work at has different rules to be followed that, over time, make sense, and as you leave the company, you take some of those rules with you as your own “best practices.”

But there’s no one size fits all in software development. Everything that we do is very opinionated and works only for us. It’s a monster built from different aspects of different paradigms that helps us, and only us, to be efficient.



Everyone knows better.

Opinionated Practices

If we take a look at development practices, there are so many different ways that you can go about building software:

1. Test-Driven Development - the Holy Grail of development everyone tries to achieve.
2. Acceptance Test-Driven Development - a variation of the TDD but more focused on explaining what needs to be done from the business perspective.
3. Behavior-Driven Development - another variation that focuses on creating a shared understanding of a description of how part of the software should work.
4. Example-Drive Development and Story TDD - even more variations of how you can define what software needs to do and how to test it.

As you can see, each one is a highly opinionated practice, and each has developers who preach it's the only proper way to build software. If you're not doing TDD, are you even doing software development?



Opinionated Frameworks

The javascript world is full of wonders. There are frameworks that all claim a different approach to DOM rendering and encapsulation of modules. You can build your web application (or even mobile application) in a thousand different ways, and each of those frameworks you use has its own best practices that do not match.

Server-side rendering got exiled not long ago, but now it's back on the table as the new cool thing with partial server-side rendering to "speed up" your application. The best practice now is to render content that changes less often server-side and the parts that often change on the browser. Some will agree, the others will say this is crazy.



SHENCOMIX.COM

Opinionated Frameworks tell you how to do things, and if you agree with them, you'll be much more efficient.

TailwindCSS is an example of a very opinionated CSS framework where the styles are done via the classes. Many people haaaaate the idea and argue, "HTML gets bloated, tailwind is like inline styles, it violates the separation of style and markup." The conventional wisdom is the opposite of Tailwind — hide as much CSS in the stylesheets and use a single class for an element. But I like the concept and think it makes me efficient when I do HTML changes for my blog, for example.

So who's right? Nobody. It's just our opinions, they're valid even if we have opposite ones.

Opinionated Products

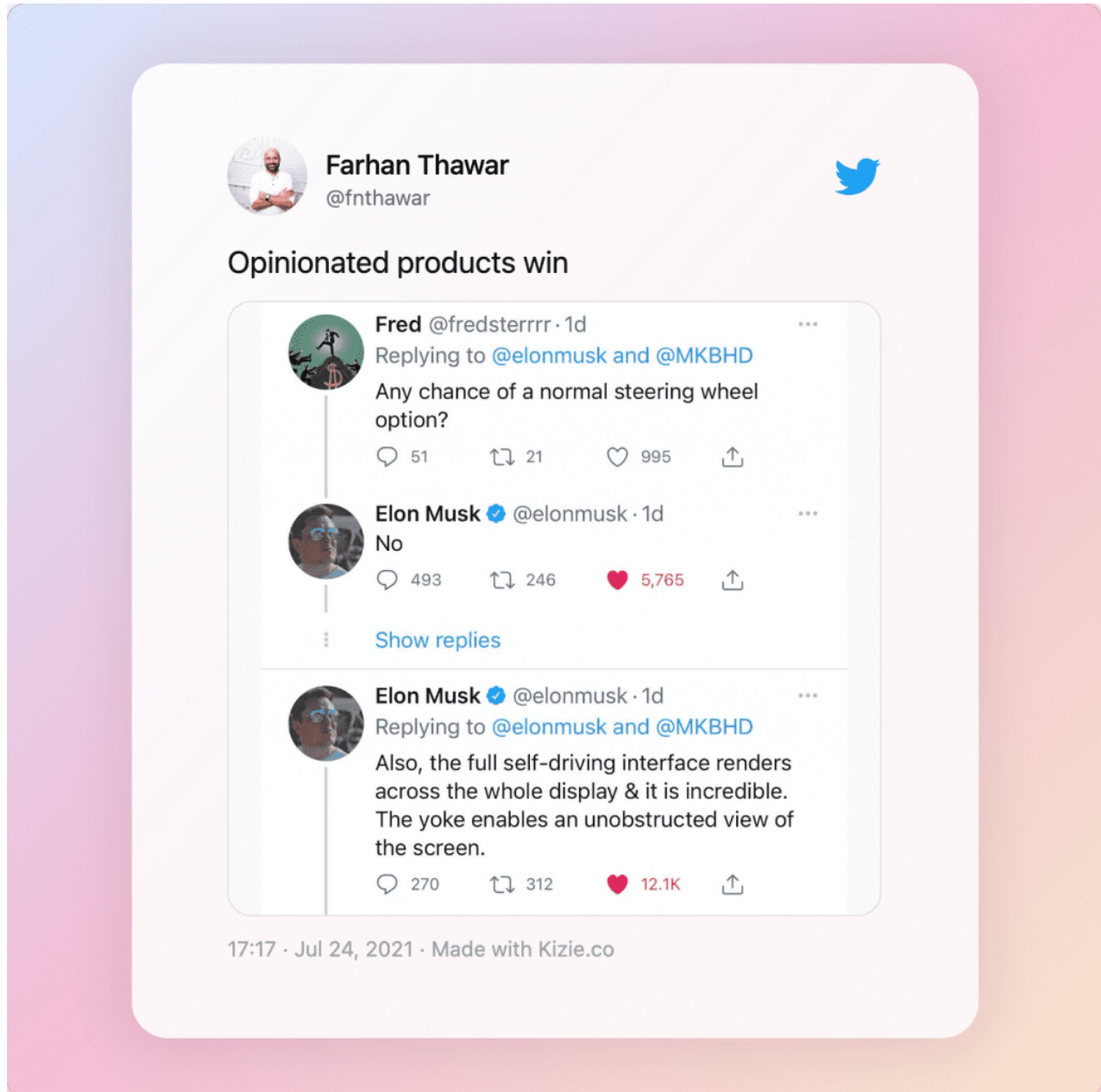
You know how Microsoft Excel tries to guess what you're trying to do, and sometimes it assumes wrong, so you have to go to the configuration and change the font, table, etc. That's convention over configuration — meaning the product works delightfully for 90% of cases but allows the other 10% to be changed when necessary.

There are a lot of opinionated products — even if we take just the calendar applications. There are so many different ways you can display calendars and weekly schedules that hundreds of applications are built with strong opinions. For example, some allow booking slots only in 1-hour blocks, and others allow only a weekly view.

An excellent example in the Java world is the Ant vs. Maven. These are build tools used to create java executables. Ant gives you the flexibility, and you need to specify everything, while Maven hides most of the stuff but allows you to override the defaults.

Maven became more popular than Ant in the Java ecosystem because it was easy to use 90% of the time, and the other 10% you can change with configuration.

Conventions create an opinionated product. Opinions create user delight. User delight creates successful businesses.



Opinionated Processes

You might think that if you're not doing agile to the letter, you're a failure. Are you even doing software development if you can't do it properly as the agile gods have decreed?

Well, that's also wrong. There are so many opinions on doing agile that the original manifesto no longer makes sense. For example, during the last few months, I've heard several contradicting statements on doing estimations and planning for agile teams.

1. You should estimate in points.
2. You should estimate in hours.
3. You should not estimate at all and break down the story into the smallest possible tasks.
4. Don't do agile at all. Waterfall is the best.

Even though you might think that "doing agile" is the best practice, there are so many opinions on how to do it that it doesn't make sense to follow any advice blindly.



Jeff Sutherland, Inventor and Co-Creator of Scrum

Answered 3 years ago

Estimating tasks will slow you down. Don't do it We gave it up over 10 years ago.

Today we have good data from Rally on 60,000 teams. The slowest estimate tasks in hours. No estimation at all will improve team performance over hour estimation.

Best teams have small stories and do no tasking. They move to acceptance test driven development.

There's a simple guide for adopting anything: you take any methodology that you think fits your goals and squeeze it, stretch it, squash it so that it works for your organization — so you can be as effective as possible. If anything inside the methodology makes you inefficient, throw it out and define your rules.



Opinionated Conclusion

There is no silver bullet solution out there that will make you automatically efficient. Quite the opposite, if you take any methodology and try to implement it exactly as it's described, your efficiency will likely drop.

Being agile does not mean having a rule for everything and just following the red dotted line. Getting shit done is the only factor that matters. And even if you have a unique adaptation of the SCRUM methodology in your company — that's also fine, as long as it works for you.

Opinions matter. The context surrounding you makes your situation unique, and the tools you select must also be adapted to your circumstances.

Every framework, programming language, and IDE you choose is just a tool that should help you get from point A to point B in an efficient way without compromising on your values. As long as you have that, then congratulations, you have a working system.

